Lecture 8: Exploration

CS234: RL

Emma Brunskill

Spring 2017

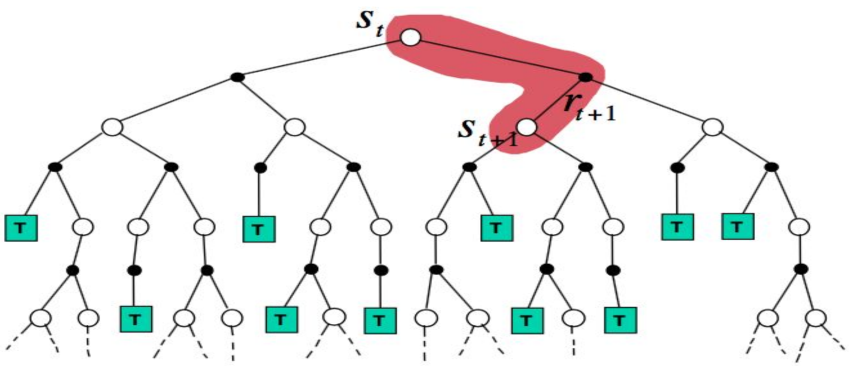
Much of the content for this lecture is borrowed from Ruslan Salakhutdinov's class, Rich Sutton's class and David Silver's class on RL.

Today

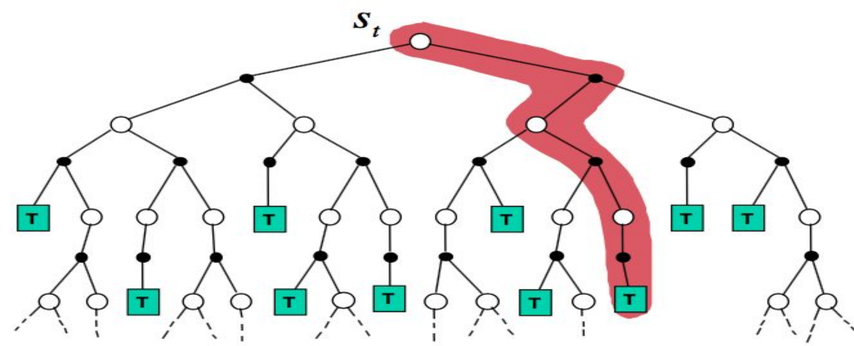
- Model free Q learning + function approximation
- Exploration

TD vs Monte Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$



$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$$



TD Learning vs Monte Carlo: Linear VFA Convergence Point

- Linear VFA: $V(s) = \sum_i w_i f_i(s)$
- Monte Carlo estimate:
- $$MSVE(w_{MC}) = \min_w \sum_{s \in S} d(s) \left(V^\pi(s) - \tilde{V}^\pi(s, w) \right)^2$$
- TD converges to constant factor of best MSE

$$MSVE(w_{TD}) = \frac{1}{1-\gamma} MSVE(w_{MC})$$

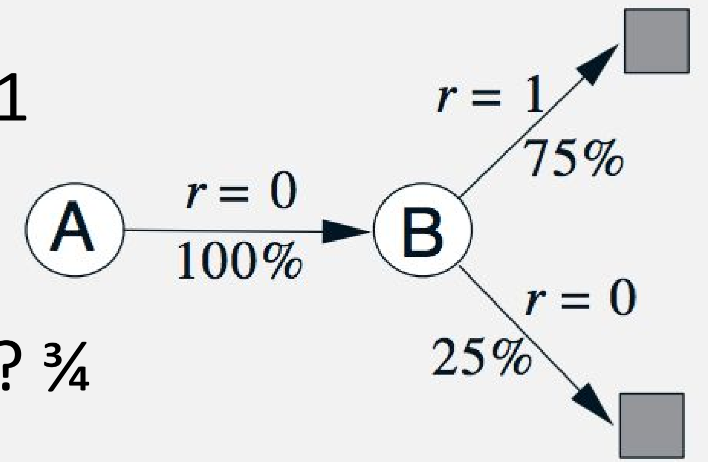
- In look up table representation, both have 0 error

Tsitsiklis and Van Roy. An Analysis of Temporal-Difference Learning with Function Approximation. 1997

TD Learning vs Monte Carlo:

Finite Data, Lookup Table, Which is Preferable?

- 8 episodes, all of 1 or 2 steps duration
  - 1st episode: A, 0, B, 0
  - 6 episodes where observe: B, 1
  - 8th episode: B, 0
- Assume discount factor = 1
- What is a good estimate for V(B)? ¾
- What is a good estimate of V(A)?
  - Monte Carlo estimate: 0
  - TD learning w/infinite replay: ¾
    - Computes certainty equivalent MDP
    - MC has 0 error on training set
    - But expect TD to do better-- **leverages Markov structure**



Example 6.4, Sutton and Barto

TD Learning & Monte Carlo: Off Policy

- In Q-learning follow one policy while learning about value of optimal policy
- How do we do this with Monte Carlo estimation?
  - Recall that in MC estimation, just average sum of future rewards from a state
  - Assumes always following same policy
- Solution for off policy MC: Importance Sampling!

Example 6.4, Sutton and Barto

Importance Sampling

- Episode/history = (s,a,r,s',a',r',s''...) (sequence of all states, actions, rewards for the whole episode)
- Assume have data from one* policy π_b
- Want to estimate value of another π_e
- First recall MC estimate of value of π_b

$$= \frac{1}{N}\sum_{j=1}^{N} R_j \qquad R_j = \sum_{i=1}^{H} \gamma^{i-1} r_{ij}$$

  - where j is the jth episode sampled from π_b

- jth history/episode= $(s_{1j}, a_{1j}, r_{1j}, s_{2,j}, a_{2,j}, r_{2,j}, \ldots) \sim \pi_b$ $R_j = \sum_{i=1}^{H} \gamma^{i-1} r_{ij}$

$$V^{\pi_e} = \int_{history} p(history|\pi_e) \left[\sum_{i=1}^{H} \gamma^{i-1} r_i | history \right]$$

- jth history/episode= $(s_{1j}, a_{1j}, r_{1j}, s_{2,j}, a_{2,j}, r_{2,j}, \ldots) \sim \pi_b$ $\quad R_j = \sum_{i=1}^{H} \gamma^{i-1} r_{ij}$

$$
\begin{aligned}
V^{\pi_e} &= \int_{history} p(history|\pi_e) \left[\sum_{i=1}^{H} \gamma^{i-1} r_i | history \right] \\
&= \int_{history} p(history|\pi_e) \frac{p(history|\pi_b)}{p(history|\pi_b)} \left[\sum_{i=1}^{H} \gamma^{i-1} r_i | history \right] \\
&= \int_{history} p(history|\pi_b) \frac{p(history|\pi_e)}{p(history|\pi_b)} \left[\sum_{i=1}^{H} \gamma^{i-1} r_i | history \right] \\
&\approx \frac{1}{N} \sum_{j=1}^{N} \frac{p(history_j|\pi_e)}{p(history_j|\pi_b)} \left[\sum_{i=1}^{H} \gamma^{i-1} r_i | history_j \right] \\
&\approx \frac{1}{N} \sum_{j=1}^{N} \frac{p(history_j|\pi_e)}{p(history_j|\pi_b)} R_j
\end{aligned}
$$

- jth history/episode= (s_{1j},a_{1j},r_{1j},$s_{2,j}$,$a_{2,j}$,$r_{2,j}$,...) ~ π_b $R_j = \sum_{i=1}^{H} \gamma^{i-1} r_{ij}$

$$
\begin{aligned}
V^{\pi_e} &= \int_{history} p(history|\pi_e) \left[\sum_{i=1}^{H} \gamma^{i-1} r_i | history \right] \\
&\approx \frac{1}{N} \sum_{j=1}^{N} \frac{p(history_j|\pi_e)}{p(history_j|\pi_b)} R_j \\
&= \frac{1}{N} \sum_{j=1}^{N} \left[\Pi_{i=1}^{N-1} \frac{p(s_{i+1}|s_i, a) p(a|\pi_e, s_i)}{p(s_{i+1}|s_i, a) p(a|\pi_b, s_i)} \right] R_j
\end{aligned}
$$

Importance Sampling

- Episode/history = (s,a,r,s',a',r',s''...) (sequence of all states, actions, rewards for the whole episode)
- Assume have data from one* policy π_b
- Want to estimate value of another π_e
- Unbiased* estimator of $\boldsymbol{\pi_e}$

$$\frac{1}{N}\sum_{j=1}^{N}\left[\Pi_{i=1}^{N-1}\frac{p(a|\pi_e,s_i)}{p(a|\pi_b,s_i)}\right]R_j$$

  - where j is the jth episode sampled from $\boldsymbol{\pi_b}$

e.g. Mandel, Liu, Brunskill, Popovic AAMAS 2014

- Need same support: if $p(a|\pi_e,s)>0$, then $p(a|\pi_b,s)>0$

TD Learning & Monte Carlo: Off Policy

- With lookup table representation
  - Both Q-learning and Monte Carlo estimation (with importance sampling) will converge to value of optimal policy
  - Requires mild conditions over behavior policy (e.g. infinitely visiting each state--action pair is one sufficient condition)
- What about with function approximation?

Example 6.4, Sutton and Barto

TD Learning & Monte Carlo: Off Policy

- With lookup table representation
  - Both Q-learning and Monte Carlo estimation (with importance sampling) will converge to value of optimal policy
  - Requires mild conditions over behavior policy (e.g. infinitely visiting each state--action pair is one sufficient condition)
- What about with function approximation?
  - Target update is wrong
  - Distribution of samples is wrong

Example 6.4, Sutton and Barto

TD Learning & Monte Carlo: Off Policy

- With lookup table representation
  - Both Q-learning and Monte Carlo estimation (with importance sampling) will converge to value of optimal policy
  - Requires mild conditions over behavior policy (e.g. infinitely visiting each state--action pair is one sufficient condition)
- What about with function approximation?
  - Q-learning with function approximation can diverge
    - See examples in Chapter 11 (Sutton and Barto)
  - But in practice often does very well

Example 6.4, Sutton and Barto

Summary: What You Should Know

- Deep learning for model-free RL
  - Understand how to implement DQN
  - 2 challenges solving and how it solves them
  - What benefits double DQN and dueling offer
  - Convergence guarantees
- MC vs TD
  - Benefits of TD over MC
  - Benefits of MC over TD

Today

- Model-free Q learning + function approximation
- Exploration

Only Learn About Actions Try

- Reinforcement learning is censored data
  - Unlike supervised learning
- Only learn about reward (& next state) of actions try
- How balance
  - exploration -- try new things that might be good
  - exploitation -- act based on past good experiences
- Typically assume tradeoff
  - May have to sacrifice immediate reward in order to explore & learn about potentially better policy

Do We Really Have to Tradeoff? (when/why?)

- Reinforcement learning is censored data
  - Unlike supervised learning
- Only learn about reward (& next state) of actions try
- How balance
  - exploration -- try new things that might be good
  - exploitation -- act based on past good experiences
- Typically assume tradeoff
  - May have to sacrifice immediate reward in order to explore & learn about potentially better policy

Performance of RL Algorithms

- Convergence
- Asymptotically optimal
- Probably approximately correct
- Minimize / sublinear regret

Performance of RL Algorithms

- Convergence
  - In limit of infinite data, will converge to a fixed V
- Asymptotically optimal
- Probably approximately correct
- Minimize / sublinear regret

Performance of RL Algorithms

- Convergence
- Asymptotically optimal
  - In limit of infinite data, will converge to optimal π
  - E.g. Q-learning with e-greedy action selection
  - Says nothing about finite-data performance
- Probably approximately correct
- Minimize / sublinear regret

Probably Approximately Correct RL

- Given an input ε and δ, with probability at least 1-δ
- On all but N steps,
- Select action a for state s whose value is ε-close to V*
  $|Q(s,a) - V^*(s)| < \varepsilon$
- where N is a polynomial function of $(|S|,|A|,\delta,\varepsilon,\Upsilon)$

- Much stronger criteria
  - Bounding number of mistakes we make
  - Finite and polynomial

Can We Use e'-Greedy Exploration to get a PAC Algorithm?

- Need eventually to be taking bad actions only small fraction of the time
- Bad (random) action could yield poor reward on this and many future time steps
- If want PAC MDP algorithm using e'-greedy exploration, need e' < $\varepsilon(1-\Upsilon)$
  - Want |Q(s,a) - V*(s)| < ε
  - Can construct cases where bad action can cause agent to incur poor reward for awhile
  - A.Strehl's PhD thesis 2007, chp 4
-

Q-learning with e'-Greedy Exploration* is not PAC

- Need eventually to be taking bad actions only small fraction of the time
- Bad (random) action could yield poor reward on this and many future time steps
- If want PAC MDP algorithm using e'-greedy exploration, need e' < $\varepsilon(1-\Upsilon)$
- *Q-learning with optimistic initialization & learning rate = (1/t) and e'-greedy exploration is not PAC
  - Even though will converge to optima
  - Thm 10 in A.Strehl thesis 2007

Certainty Equivalence with e'-Greedy Exploration* is not PAC

- Need eventually to be taking bad actions only small fraction of the time
- Bad (random) action could yield poor reward on this and many future time steps
- Q-learning with optimistic initialization & learning rate = (1/t) and e'-greedy exploration is not PAC
- *Certainty euivalence model-based RL w/ optimistic initialization and e-greedy exploration is not PAC
  - A.Strehl's PhD thesis 2007, chp 4, theorem 11

e'-Greedy Exploration has not been shown to yield PAC MDP RL

- So far (to my knowledge) no positive results that can make at most a polynomial # of time steps where may select non-ε optimal action
- But interesting open issue and there is some related work that suggests this might be possible
  - Could be a good theorey CS234 project!
  - Come talk to me if you're interested in this

PAC RL Approaches

- Typically model-based or model free
- Formally analyze how much experience is needed in order to estimate a good Q function that we can use to achieve high reward in the world

Good Q $\longrightarrow$ Good Policy

- Homework 1 quantified how if have good (e-accurate) estimates of the Q function, can use to extract a policy with a near optimal value

PAC RL Approaches: Model-based

- Formally analyze how much experience is needed in order to estimate a good model (dynamics and reward models) that we can use to achieve high reward in the world

“Good” RL Models

- Estimate model parameters from experience
- More experience means our estimated model parameters will closer be to the true unknown parameters, with high probability

Acting Well in the World

$p(s' \mid s, a)$ known $\rightarrow$ Compute ε-optimal policy

Bound $\left(\hat{p}(s' \mid s, a) - p(s' \mid s, a)\right) \rightarrow$ Bound error in policy calculated using $\hat{p}(s' \mid s, a)$

How many samples do we need to build a good model that we can use to act well in the world?

Sample complexity	=	# steps on which may not act well (could be far from optimal)
(R-MAX and E^3)	=	Poly(# of states)

PAC RL

- If e’-greedy is insufficient, how should we act to achieve PAC behavior (finite # of potentially bad decisions)?

Proposition 1 *Let $\mathcal{A}(\epsilon, \delta)$ be any greedy learning algorithm such that for every timestep t, there exists a set K_t of state-action pairs. We assume that $K_t = K_{t+1}$ unless, during timestep t, an update to some action value occurs or the event A_K happens. Let M_{K_t} be the known state-action MDP and π_t be the current greedy policy, that is, for all states s, $\pi_t(s) = \text{argmax}_a Q_t(s, a)$. Suppose that for any inputs ϵ and δ, with probability at least $1 - \delta$, the following conditions hold for all states s, actions a, and timesteps t: (1) $Q_t(s, a) \geq Q^*(s, a) - \epsilon$ (optimism), (2) $V_t(s) - V^{\pi_t}_{M_{K_t}}(s) \leq \epsilon$ (accuracy), and (3) the total number of updates of action-value estimates plus the number of times the escape event from K_t, A_K, can occur is bounded by $\zeta(\epsilon, \delta)$ (learning complexity). Then, when $\mathcal{A}(\epsilon, \delta)$ is executed on any MDP M, it will follow a 4ϵ-optimal policy from its current state on all but*

$$O\left(\frac{\zeta(\epsilon, \delta)}{\epsilon(1-\gamma)^2} \ln \frac{1}{\delta} \ln \frac{1}{\epsilon(1-\gamma)}\right)$$

Sufficient Condition for PAC Model-based RL

Optimism under uncertainty!

Strehl, Li, Littman 2006

Sufficient Condition for PAC Model-based RL

Proposition 1 *Let $\mathcal{A}(\epsilon, \delta)$ be any greedy learning algorithm such that for every timestep t, there exists a set K_t of state-action pairs. We assume that $K_t = K_{t+1}$ unless, during timestep t, an update to some action value occurs or the event A_K happens. Let M_{K_t} be the known state-action MDP and π_t be the current greedy policy, that is, for all states s, $\pi_t(s) = \operatorname{argmax}_a Q_t(s, a)$. Suppose that for any inputs ϵ and δ, with probability at least $1 - \delta$, the following conditions hold for all states s, actions a, and timesteps t: (1) $Q_t(s, a) \geq Q^*(s, a) - \epsilon$ (optimism), (2) $V_t(s) - V^{\pi_t}_{M_{K_t}}(s) \leq \epsilon$ (accuracy), and (3) the total number of updates of action-value estimates plus the number of times the escape event from K_t, A_K, can occur is bounded by $\zeta(\epsilon, \delta)$ (learning complexity). Then, when $\mathcal{A}(\epsilon, \delta)$ is executed on any MDP M, it will follow a 4ϵ-optimal policy from its current state on all but*

$$O\left(\frac{\zeta(\epsilon, \delta)}{\epsilon(1-\gamma)^2} \ln \frac{1}{\delta} \ln \frac{1}{\epsilon(1-\gamma)}\right)$$

Optimism under uncertainty!

Strehl, Li, Littman 2006

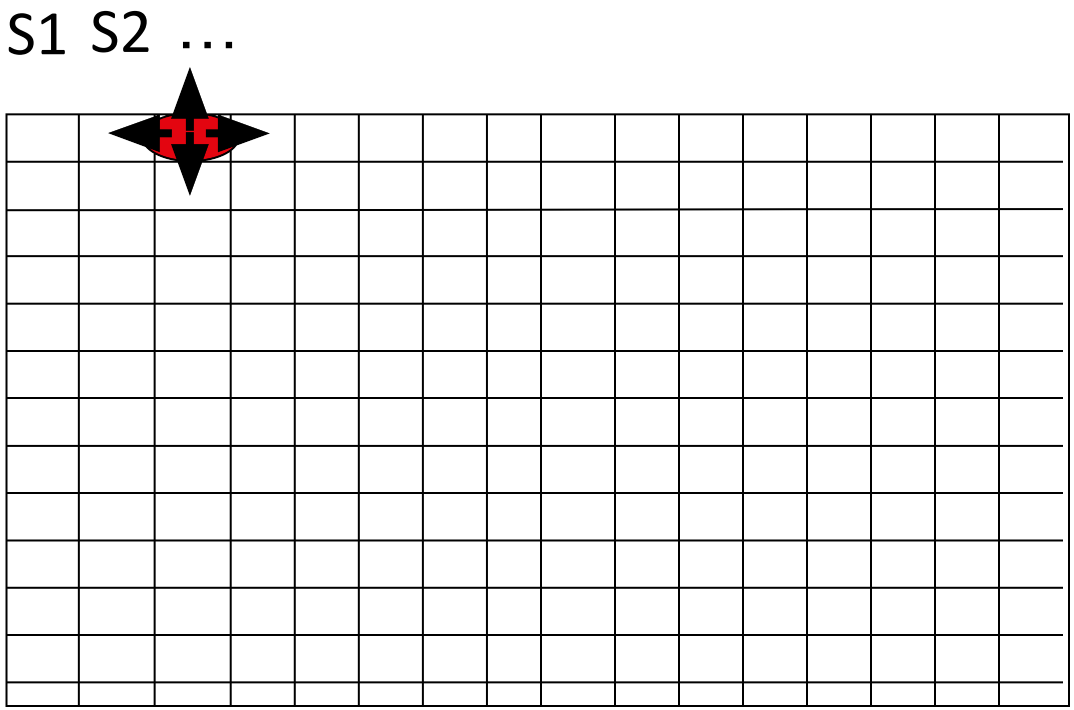
Important Ideas in PAC RL

- Bound error over model estimates
  - Relate amount of samples to accuracy of parameters
- Be optimistic with respect to model / Q uncertainty
  - Consider how world could be
  - Solve policy for that world
  - Act accordingly

Model-Based RL

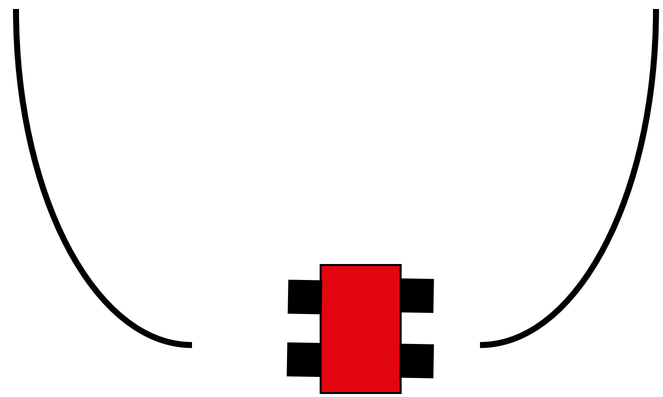
- Given data seen so far
- Build an explicit model of the MDP
- Compute policy for it
- Select action for current state given policy, observe next state and reward
- Repeat

R-max (Brafman & Tennenholtz)



R-max is Model-based RL

Think hard: estimate models & compute policies



Act in world

Rmax leverages optimism under uncertainty!

R-max Algorithm: Initialize: Define "Known" MDP

Reward

Known/ Unknown

	S1	S2	S3	S4	…
↑	U	U	U	U	
→	U	U	U	U	
↓	U	U	U	U	
←	U	U	U	U	

	S1	S2	S3	S4	…
↑	R_{max}	R_{max}	R_{max}	R_{max}	
→	R_{max}	R_{max}	R_{max}	R_{max}	
↓	R_{max}	R_{max}	R_{max}	R_{max}	
←	R_{max}	R_{max}	R_{max}	R_{max}	

Transition Counts

	S1	S2	S3	S4	…
↑	0	0	0	0	
→	0	0	0	0	
↓	0	0	0	0	
←	0	0	0	0	

In the "known" MDP, any unknown (s,a) pair has its dynamics set as a self loop & reward = Rmax

R-max Algorithm

Plan in known MDP

R-max: Planning

- Compute optimal policy π_{known} for "known" MDP

Exercise: What Will Initial Value of Q(s,a) be for each (s,a) Pair in the Known MDP? What is the Policy?

Known/ Unknown

	S1	S2	S3	S4	…
↑	U	U	U	U	
→	U	U	U	U	
↓	U	U	U	U	
←	U	U	U	U	

Reward

	S1	S2	S3	S4	…
↑	R_{max}	R_{max}	R_{max}	R_{max}	
→	R_{max}	R_{max}	R_{max}	R_{max}	
↓	R_{max}	R_{max}	R_{max}	R_{max}	
←	R_{max}	R_{max}	R_{max}	R_{max}	

Transition Counts

	S1	S2	S3	S4	…
↑	0	0	0	0	
→	0	0	0	0	
↓	0	0	0	0	
←	0	0	0	0	

In the “known” MDP, any unknown (s,a) pair has its dynamics set as a self loop & reward = Rmax

R-max Algorithm

Act using policy

Plan in known MDP

- Given optimal policy π_{known} for "known" MDP
- Take best action for current state $\pi_{known}(s)$, transition to new state s' and get reward r

R-max Algorithm

Act using policy

Plan in known MDP

Update state-action counts

Update Known MDP

Known/
Unknown

	S2	S2	S3	S4	…
↑	U	U	U	U	
→	U	U	U	U	
↓	U	U	U	U	
←	U	U	U	U	

Reward

	S2	S2	S3	S4	…
↑	R_{max}	R_{max}	R_{max}	R_{max}	
→	R_{max}	R_{max}	R_{max}	R_{max}	
↓	R_{max}	R_{max}	R_{max}	R_{max}	
←	R_{max}	R_{max}	R_{max}	R_{max}	

Transition
Counts

	S2	S2	S3	S4	…
↑	0	0	0	0	
→	0	0	1	0	
↓	0	0	0	0	
←	0	0	0	0	

Increment counts for
state-action tuple

Update Known MDP

Known/
Unknown

	S2	S2	S3	S4	…
↑	U	U	U	U	
→	U	U	**K**	U	
↓	U	U	U	U	
←	U	U	U	U	

Reward

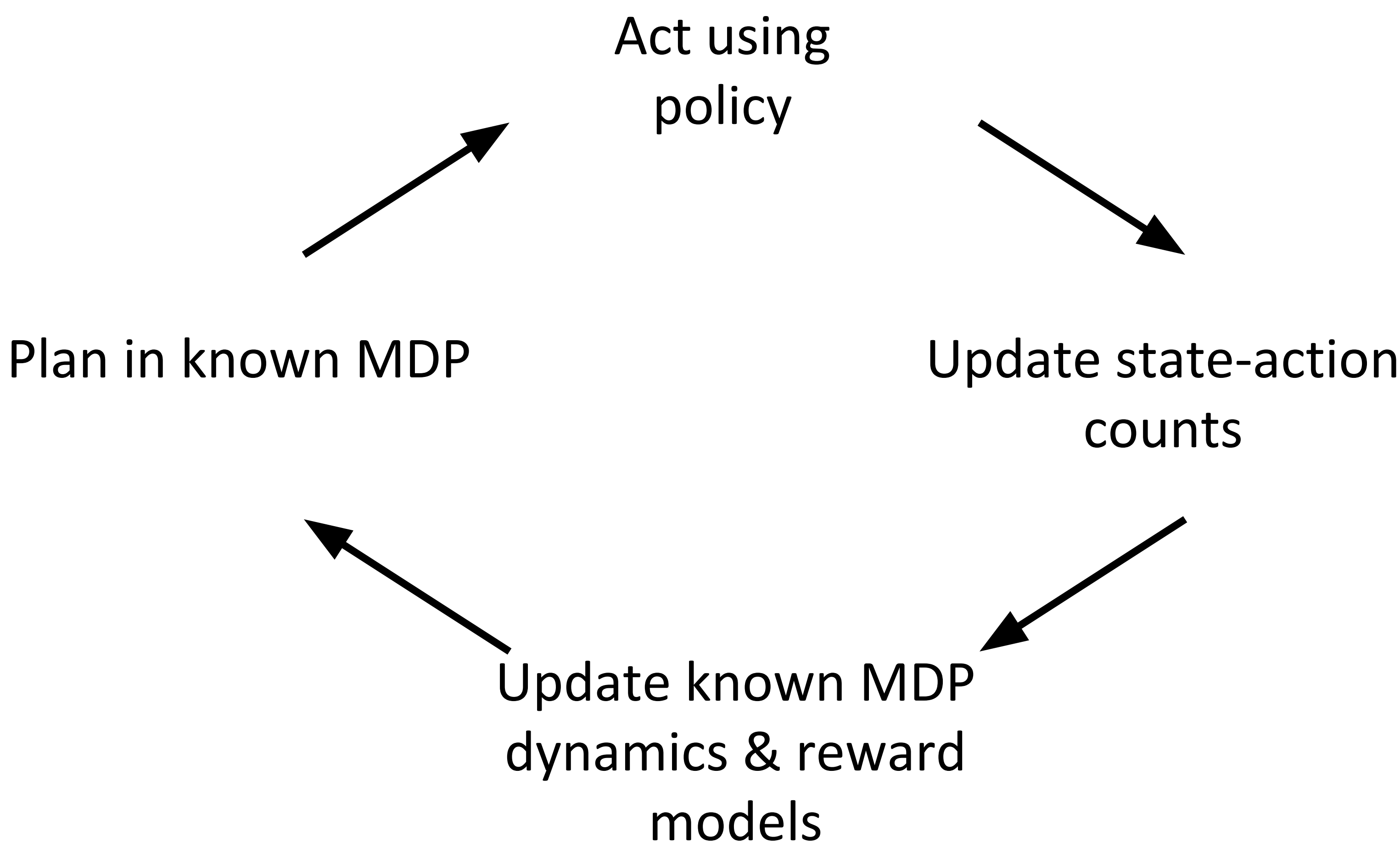
	S2	S2	S3	S4	…
↑	R_{max}	R_{max}	R_{max}	R_{max}	
→	R_{max}	R_{max}	**R**	R_{max}	
↓	R_{max}	R_{max}	R_{max}	R_{max}	
←	R_{max}	R_{max}	R_{max}	R_{max}	

Transition
Counts

	S2	S2	S3	S4	…
↑	3	3	4	3	
→	2	4	**5**	0	
↓	4	0	4	4	
←	2	2	4	1	

If counts for (s,a) > N, (s,a) becomes known: **use observed data to estimate transition & reward model** for (s,a) when planning

R-max Algorithm



Important Ideas in PAC RL

- Bound error over model estimates
  - Relate amount of samples to accuracy of parameters
- Be optimistic with respect to model / Q uncertainty
  - Consider how world could be
  - Solve policy for that world
  - Act accordingly
  - Why is that a good idea?

Sample Complexity of R-max

$$\left(\frac{SA}{\varepsilon(1-\gamma)^2} \underbrace{\frac{S}{\varepsilon^2(1-\gamma)^4}}_{\substack{\text{\# samples} \\ \text{need per (s,a)} \\ \text{pair}}} \right)$$

On all but the above number of steps, chooses action whose expected reward is close to expected reward of action take if knew model parameters, with high probability

Sample Complexity of R-max

$$\left(\frac{SA}{\varepsilon(1-\gamma)^2} \underbrace{\frac{S}{\varepsilon^2(1-\gamma)^4}}_{} \right)$$

γ=.9, ε=.1
How many steps?

samples
need per (s,a)
pair

On all but the above number of steps, chooses action whose expected reward is close to expected reward of action take if knew model parameters, with high probability